From: N.K. Denzin, Y.S. Lincoln 2003
The Landscape of Qualitative Research
London: Sage

8

Software and Qualitative Research

Eben A. Weitzman

◆ The array of software available to support the work of qualitative researchers is maturing. A wide variety of useful tools are now available to support many different approaches to qualitative research. Most qualitative researchers can now find software that is appropriate to their analysis plans, the structure of their data, and their ease-of-use and cost preferences. However, making that appropriate match still requires systematic analysis of the needs of the project and the researcher(s), and careful comparison of the software options available at the time of purchase with an eye kept fixed firmly on those needs. There is still no one best program.

To help researchers understand what software can and cannot do to support their research efforts, understand both the potential benefits and pitfalls of using computers in qualitative research projects, and find software that is suited to their needs, I provide in this chapter (a) an introduction to and overview of the role of software in qualitative research, (b) a discussion of the critical debates and concerns in the field about the impact and appropriateness of using qualitative data analysis (QDA) software, (c) guidelines for choosing software to match individual needs, and

AUTHOR'S NOTE: My thanks to Norman Denzin, Nigel Fielding, Udo Kelle, Ray Lee, and Morten Levin for their comments on an earlier draft of this chapter.

(d) an indication of future directions for both scholarship on the use of QDA software and development of such software.

◆ A Minihistory of theUse of Computers in Qualitative Research

Traditionally, qualitative researchers have carried out the mechanics of analysis by hand: typing up field notes and interviews, photocopying them, "coding" by marking them up with markers or pencils, cutting and pasting the marked segments onto file cards, sorting and shuffling cards, and typing up their analyses. This picture has been slowly changing since the early to mid-1980s. At that point, some researchers were beginning to use word processors for the typing work, and just a few were beginning to experiment with database programs for storing and accessing their texts. Most qualitative methods textbooks at the time (e.g., Bogdan & Biklen, 1982; Goetz & LeCompte, 1984; Lofland & Lofland, 1984; Miles & Huberman, 1984) made little, if any, reference to the use of computers.

In the early 1980s, a couple of programs designed specifically for the analysis of qualitative data began to appear (Drass, 1980; Seidel & Clark, 1984; Shelly & Sibert, 1985). Early programs like QUALOG and the first versions of The Ethnograph and NUD•IST reflected the state of computing at that time. Researchers typically accomplished the coding of texts (tagging chunks of text with labels—codes—that indicate the conceptual categories the researcher wants to sort them into) by typing in line numbers and code names at a command prompt, and there was little or no facility for memoing or other annotation or markup of text. In comparison to marking up text with colored pencils, this felt awkward to many researchers. And computer support for the analysis of video or audio data was at best a fantasy.

But the landscape has changed dramatically, in terms of both software and the literature devoted to it. By the time the late Matt Miles and I wrote *Computer Programs for Qualitative Data Analysis* (Weitzman & Miles, 1995b), we reviewed no fewer than 24 different programs that were useful for analyzing qualitative data. Half of those programs had been developed specifically for qualitative data analysis, whereas the other half had been developed for more general-purpose applications, such as text search and storage. Since then, the field has continued to grow rapidly. Programs are being revised at a regular rate, new programs appear on the scene at the

rate of one or two a year, and programs that don't find users disappear. There has been some convergence as good features in one program are imitated by the developers of others. And there has also been divergence, as developers look for new and different ways to conceptualize support for analysis.

There are now tools available that can help researchers who are using a wide variety of research and analysis methodologies, from grounded theory to textual analysis to narrative analysis to interpretive interactionism. It is important to emphasize that software is not now, if it ever was, something that is relevant only to "positivist" or "quasi-positivist" approaches to qualitative research. If you see language in this chapter that does not match your approach, you may find it helpful to do some speculative translation. For example, if the discussion is about "verification" or "hypothesis testing" and your approach is postmodern, the discussion may seem irrelevant. But it may be that there is a way to understand the concept that makes sense from your perspective, such as "looking to see whether there is more material supporting, or contradicting, a certain assumption or interpretation." The same software tools that someone else might use for classical hypothesis testing might be very useful for *your* purposes.

Many programs now allow the researcher to specify relationships among codes and use these relationships in analysis, and to write memos and link them to text and codes. Some programs allow the researcher to create links between different points in the text (hypertext), and a small but growing handful allow the use of audio and video in place of, or in addition to, text. And there are a variety of approaches to linking categorical and quantitative data (e.g., demographics, test scores, quantitative ratings) to text and for exporting categorical and quantitative data (e.g., word frequencies or coding summaries) to quantitative analysis programs for statistical analysis. Finally, there are now some free programs available, notably two from the U.S. Centers for Disease Control: EZ-Text, which focuses on qualitative surveys, and AnSWR, intended for a more general range of qualitative data. The software continues to vary widely, and it remains very much the case that there is no one best program for all needs.

In parallel with the growth in software, literature reporting studies and commenting on the software has begun to appear regularly. There has been an outpouring of journal articles, a series of international conferences on computers and qualitative methodology, thoughtful books on the

topic (Fielding & Lee, 1991, 1998; Kelle, 1995; Tesch, 1990; Weitzman & Miles, 1995b), and special journal issues (Mangabeira, 1996; Tesch, 1991).

Periodically, commentators have raised concerns about whether the range of available software is dominated by a particular approach, methodology, or epistemology (see, e.g., Coffey, Holbrook, & Atkinson, 1996; Lonkila, 1995). Although there is certainly room for further development to support certain specific analytic processes (I offer some suggestions later in this chapter, and the list appearing in the chapter titled "Reflections and Hopes" in Weitzman & Miles, 1995b, has only begun to be addressed), these concerns are clearly missing the mark. In this chapter, I suggest a wide variety of types of programs that are available to support a wide variety of research approaches. Qualitative researchers are not limited only to coding-oriented programs, or even to programs explicitly marketed to qualitative researchers. For example, as Fielding and Lee (1998) point out, there are a variety of options for those wishing to follow the suggestion of Coffey and Atkinson (1996) that text retrievers may be more helpful for discourse analysis than code-and-retrieve programs. Fielding and Lee go on to argue that

> developers of CAQDAS [computer-aided qualitative data analysis software] programs have increasingly included facilities for proximity searching, which might be useful for narrative analysis, and for "autocoding" which could be adapted to some kinds of semiotic analysis. The provision of new features in CAQDAS programs reflects the generally close relationship between users and developers characteristic of the field, and the general willingness of developers to incorporate features desired by users even if these do not always accord with the epistemological preferences of the developer. Since packages increasingly support procedures, routines and features which are new to qualitative analysis or make procedures possible that were not practicable without the power of the computer, it is less and less plausible either to argue that the software is merely an aid to code-and-retrieve or to argue that code-and-retrieve is the *sine qua non* of qualitative analysis. (p. 175)

I address these issues at more length through much of this chapter, particularly in the subsections below headed "False Hopes and Fears," "Real Hopes," and "Real Fears," and in the later section headed "Debates in the Field."

◆ What Software Can and Cannot Do

Simply put, software can provide tools to help you analyze qualitative data, but it cannot do the analysis for you, not in the same sense in which a statistical package like SPSS or SAS can do, say, multiple regression. Many researchers have had the hope—for others it is a fear—that the computer could somehow read the text and decide what it all means. That is, generally speaking, not the case.[1] Thus it is particularly important to emphasize that using software cannot be a substitute for learning data analysis methods: The researcher must know what needs to be done, and do it. The software provides tools to do it with.

The following are some of the things computers *can* be used for to facilitate the analysis process:

1. *Making notes* in the field;
2. *Writing up* or transcribing field notes;
3. *Editing:* correcting, extending, or revising field notes;
4. *Coding:* attaching key words or tags to segments of text, graphics, audio, or video to permit later retrieval;
5. *Storage:* keeping text in an organized database;
6. *Search and retrieval:* locating relevant segments of text and making them available for inspection;
7. *Data "linking":* connecting relevant data segments to each other, forming categories, clusters, or networks of information;
8. *Memoing:* writing reflective commentaries on some aspect of the data, theory, or method as a basis for deeper analysis;
9. *Content analysis:* counting frequencies, sequences, or locations of words and phrases;
10. *Data display:* placing selected or reduced data in a condensed, organized format, such as a matrix or network, for inspection;
11. *Conclusion drawing and verification:* aiding in the interpretation of displayed data and the testing or confirmation of findings;
12. *Theory building:* developing systematic, conceptually coherent explanations of findings; testing hypotheses;
13. *Graphic mapping:* creating diagrams that depict findings or theories;
14. *Report writing:* interim and final (adapted from Miles & Huberman, 1994, p. 44).

Obviously, many of these are things that researchers can do with a word processor. Other software, which is the focus of this chapter, helps with the other tasks. The developments seen in recent years have made it possible for researchers to do these things more and more easily, and more and more powerfully. In the section headed "Types and Functions of Software for QDA," below, you will find more specific details about what software can do. But first, consider some of the hopes and fears, both real and false, that people have concerning QDA software.

False Hopes and Fears

In Weitzman and Miles (1995b), we argue:

> As Pfaffenberger . . . points out, it's equally naïve to believe that a program is (a) a neutral technical tool or (b) an overdetermining monster. The issue is understanding a program's properties and presuppositions, and how they can support or constrain your thinking to produce unanticipated effects. (p. 330)

As already mentioned, many people apparently continue to believe that QDA software intends to *do* the data analysis. Skeptical researchers raise challenges to the notion of "dumping my text into a program and seeing what comes out." Others express this more as a hope that if they buy the right program, they will not have to engage in the often very time-consuming process of analyzing all that text themselves. QDA software provides tools that help you do these things; it does not do them for you.

In an extension of this concern, many researchers have worried about the software going yet a step further and "building theory." But, as Miles and I also argued in 1995, "Software will never 'do' theory building for you . . . , but it can explicitly support your intellectual efforts, making it easier for you to think coherently about the meaning of your data" (Weitzman & Miles, 1995b, p. 330).

This situation may change in the coming years. There are some current efforts to use artificial intelligence (AI) approaches to get computers to interpret text. For example, the SPSS module TextSmart uses information about frequency of occurrence of words and proximity of words to each other to categorize text responses automatically. Microsoft Word97 has an "Auto Summarize" feature that aims to identify the most important "concepts" in a document according to word frequency. Other developers are

thinking about using AI techniques to get software to participate in the theory-building process with researchers. These approaches rely on things like frequency to indicate importance and proximity in the text to indicate relatedness. For some qualitative researchers these are acceptable assumptions, but for many others they are not, and for such researchers the results of these approaches do not yield useful interpretations of text.

Real Hopes

What can we really expect to gain from the use of software? QDA software provides tools for searching, marking up, linking, and reorganizing the data, and representing and storing your own reflections, ideas, and theorizing. Some of it gives you tools for further exploration—which in some cases might amount to hypothesis testing or conclusion verification—based on your theorizing and interpretive work.

Consistency. Software can help with consistency. If I can search for *all* the places a given key word appears, or *all* the places where a given code or combination of codes was applied, or always see the relationship between two features of the data that I have recorded, it becomes possible for me to be more consistent in a couple of ways. I can be much more careful about not missing the data that contradict my brilliant, but wrong, new hypothesis. I can easily review all the data I assigned to a given conceptual category or theme and check to see if they (a) all belong together and (b) still seem to support the interpretation I started out with; if not, I can easily reorganize. (Note that the problem of my making bad interpretations has not been removed. But the kinds of facilities mentioned here can be tremendously helpful to competent researchers in checking their own work, as well as in allowing colleagues or research participants to check it and provide feedback.)

Speed. The speed of computers is a critical issue in making QDA software helpful. First, a caution: It can take time to learn to use a program, and once you have, it can take some time to prepare and set up the data for analysis. But once that is done, the speed of the computer quickly pays for that investment. Being able to search and re-search almost instantaneously encourages the researcher to conduct multiple searches to zero in on the data that really apply to a particular question. Being able to quickly re-sort a database, redefine codes, and reassign chunks of text enables and

encourages the researcher to revise the analysis and the thinking about it whenever necessary. Being able to quickly pull together all the text for cells in a complex matrix display enables and encourages the researcher to run down provocative leads and new ideas—as well as worries that the current conclusions may be way off track—much more often and with much less cost.

An example from quantitative research may be instructive. In the days of slide rules, and even of handheld calculators, before statistical software was available, doing factor analysis was a months-long enterprise. Now a factor analysis can be run in minutes or seconds on a desktop computer. As a result, researchers can run factor analyses much more often, as part of other analyses rather than only as major undertakings of their own, and on multiple sets of scores in the same project. The speed of the computer alone can change what researchers even contemplate undertaking.

Representation. Software that allows dynamic, real-time representation of a researcher's thinking can be a substantial aid to theorizing. Software that provides a graphic map of relationships among codes, text segments, or cases can help researchers to visualize and extend their thinking about the data or theory at hand. Researchers often use drawings to depict these relationships, but software can keep maps tied to the underlying project, so that changes to the links in the drawing change the links among the objects in the database, and vice versa.

Consolidation. Finally, allowing the researcher to record field notes, interviews, codes, memos, annotations, reflective remarks, diagrams, audio and video recordings, demographic variables, and structural maps of the data and the theory all in one place can be a tremendously powerful support to the analysis process. If the design of the program is such that it allows the researcher to move from one intellectual activity to another with minimal effort, and carry over the results of one sort of thinking to others, it can both free up large amounts of energy for the critical tasks and help the researcher to see and keep track of connections that might otherwise easily fall through the cracks.

Real Fears

What do we really have to be worried about? Many of the advantages touted above have flip sides. The very ease, speed, and power of the

software have the potential to encourage the kind of thinking I have referred to as "false hopes and fears" above. Although the software will not figure out what a complex account of a childhood trauma really means in the context of the current study, the ease of searching for key words and "autocoding" them may encourage the researcher to take shortcuts. We may fail to check to see what passages were actually coded in the autocoding process, and to use their own intelligence to analyze whether they fit. There is the real potential that we will get lazy. As Lee and Fielding (1991) have noted, "There is the possibility that the use of computers may tempt qualitative researchers into 'quick and dirty' research with its attendant danger of premature theoretical closure" (p. 8).

It is also possible that the availability of software may tempt researchers to skip over the process of learning properly about research. Again from Lee and Fielding (1991):

> Of course, the ultimate fear here is of Frankenstein's monster. It is susceptible to the same caveats, too. Like the monster, the programs are misunderstood. The programs are innocent of guile. It is their misapplication which poses the threat. It was exposure to human depravity which made a threat of Frankenstein's creation. Equally, the untutored use of analysis programs can certainly produce banal, unedifying and off-target analyses. But the fault would lie with the user. This is why teaching the use of the programs to novice researchers has to be embedded in a pedagogy which has a sense of the exemplars of qualitative analysis, rather than as skills and techniques to be mechanically applied. (p. 8)

The final fear that has some truth to it is that the conceptual assumptions behind the program—for example, that the relationships among codes are always strictly hierarchical—will shape the analysis. This fear both has truth to it and is often overstated. For example, if the program allows you to directly represent hierarchical relationships among codes, but not nonhierarchical relationships, such as circular loops or unstructured networks, it will probably encourage you to think primarily or only in terms of hierarchical relationships among your codes/concepts. If you are aware of the assumptions behind a program, you have a couple of options: You can choose another program or you can find a way to work around the assumptions in the program—for example, by keeping an ever-changing, nonhierarchical code map pinned to the wall. More on this in the section headed "Debates in the Field," below.

◆ Types and Functions of Software for QDA

In this section I offer a rough sorting of available software into types. There is naturally quite a bit of overlap among categories, with individual programs having functions that would seem to belong to more than one type. However, it is possible to focus on the "heart and soul" of a program: what it is mainly intended for. This categorization scheme was first presented in Weitzman and Miles (1995b).

Text Retrievers

Text retrievers specialize in finding all the instances of words and phrases in text, in one or several files. They typically also allow you to search for places where two or more words or phrases coincide within a specified distance (a number of words, sentences, pages, and so on) and allow you to sort the resulting passages into different output files and reports. They may do other things as well, such as content analysis functions like counting, displaying key words in context or creating concordances (organized lists of all words and phrases in their contexts), or they may allow you to attach annotations or even variable values (for things like demographics or source information) to points in the text. Examples of text retrievers are Sonar Professional, the Text Collector, and ZyINDEX; there are also a variety of free (but hard to use) GREP tools available on the World Wide Web.

Textbase Managers

Textbase managers are database programs specialized for storing text in more or less organized fashion. They are good at holding text, together with information about it, and allowing you to quickly organize and sort your data in a variety of ways and retrieve it according to different criteria. Some are better suited to highly structured data that can be organized into "records" (that is, specific cases) and "fields" (variables—information that appears for each case), whereas others easily manage "free-form" text. They may allow you to define fields in the fixed manner of a traditional database such as Microsoft Access® or FileMaker Pro®, or they may allow significantly more flexibility, for example, allowing different records to have different field structures. Their search operations may be as good as, or sometimes even better than, those of some text retrievers. Examples of

textbase managers are askSam, Folio Views, Idealist, InfoTree32 XT, and TEXTBASE ALPHA.

Code-and-Retrieve Programs

Code-and-retrieve programs are often developed by qualitative researchers specifically for the purpose of qualitative data analysis. The programs in this category specialize in allowing you to apply category tags (codes) to passages of text and later retrieve and display the text according to your coding. These programs have at least some search capacity, allowing you to search either for codes or words and phrases in the text. They may have a capacity to store memos. Even the weakest of these programs represent a quantum leap forward from the old scissors-and-paper approach: they're more systematic, more thorough, less likely to miss things, more flexible, and much, much faster. Examples of code-and-retrieve programs are HyperQual2, Kwalitan, QUALPRO, Martin, and the Data Collector.

Code-Based Theory Builders

Most of the code-based theory-building programs are also based on a code-and-retrieve model, but they go beyond the functions of code-and-retrieve programs. They do not, nor would you want them to, build theory for you. Rather, they have special features or routines that go beyond those of code-and-retrieve programs in supporting *your* theory-building efforts. For example, they may allow you to represent relations among codes, build higher-order classifications and categories, or formulate and test theoretical propositions about the data. They may have more powerful memoing features (allowing you, for example, to categorize or code your memos) or more sophisticated search-and-retrieval functions than code-and-retrieve programs. They may have extended and sophisticated hyperlinking features, allowing you to link segments of text together or to create links among segments of text, graphics, photos, video, audio, Web sites, and more. They may also offer capabilities for "system closure," allowing you to feed results of your analyses (such as search results or memos) back into the system as data. Examples of code-based theory builders are AFTER, AnSWR, AQUAD, ATLAS/ti, Code-A-Text, HyperRESEARCH, NUD•IST, NVivo, QCA, the Ethnograph, and winMAX. Two of these programs, AQUAD and QCA, support cross-case configural

analysis (Ragin, 1987), QCA being dedicated wholly to this method and not having any text-coding capabilities.

Conceptual Network Builders

Conceptual network builders are programs that emphasize the creation and analysis of network displays. Some of them are focused on allowing you to create network drawings: graphic representations of the relationships among concepts. Examples of these are Inspiration, MetaDesign, and Visio. Others are focused on the analysis of cognitive or semantic networks, for example, the program MECA. Still others offer some combination of the two approaches, for example, SemNet and Decision Explorer. Finally, ATLAS/ti, a program also mentioned above under code-based theory builders, also has a fine graphical network builder connected to the analytic work you do with your text and codes.

Summary

In concluding this discussion of the five main software family types, I want to emphasize that functions often cross type boundaries. For example, Folio VIEWS can code and retrieve, and has an excellent text search facility. ATLAS/ti, NUD•IST, NVivo, the Ethnograph, and winMAX graphically represent the relationships among codes, although among these, only ATLAS/ti allows you to work with and manipulate the drawing.[2] The Ethnograph and winMAX both have systems for attaching variable values (text, date, numeric, and so on) to text files and/or cases. Sphinx Survey allows you to work with survey data consisting of a mix of qualitative and quantitative data. The implication: Do not decide too early which family you want to choose from. Instead, stay focused on the functions you need.

Multimedia. Multimedia capabilities are just beginning to emerge as a significant issue in software choice. There are now several programs in the code-based theory builder category that allow you to use audio and video, as well as text, as data: AFTER, ATLAS/ti, and Code-A-Text all allow you to code and annotate audio and video files, and search and retrieve from them, in ways quite similar to the ways they let you manipulate text, as does version 2 of HyperRESEARCH, which is under development at the time of this writing. In these programs, you can play a media file (audio or

video), mark the beginning and ending points of segments, and then treat those segments much like segments of text. A program now in beta testing called InterClipper is designed primarily for audio files, with the assumption that you only bother to transcribe the segments that you find most important (it is targeted at focus group researchers in commercial environments who need to be able to generate analyses and reports quickly). This program will probably fall in the code-and-retrieve family when it is ready for release. There is also a growing field of software dedicated exclusively to managing video.

♦ How to Make Intelligent, Individualized Software Choices

I have emphasized from the beginning of this chapter that there is no one best software program for analyzing qualitative data. Furthermore, there is no one best program for a particular type of research or analytic method. Researchers will sometimes ask, What's the best program for a school ethnography? or, What's the best program for doing grounded theory? or, What's the best program for analyzing focus groups? None of these questions has a good answer. Instead, analysts need to approach choice based on the structure of the data, the specific things they will want to do as part of the analysis, and their needs around issues such as ease of use, cost, time available, and collaboration.

Researchers can ask themselves four broad questions, as well as consider two cut-across issues, to help guide their choices (Weitzman & Miles, 1995a, 1995b). These guidelines for choice have seen wide use in practice since their original formulation, and have proven to be effective for guiding researchers to appropriate choices. Because this approach to choice emphasizes matching functions, rather than specific programs, to particular needs, these guidelines can continue to be useful long after the programs referenced here as examples have evolved into new versions and new programs have arrived on the scene.[3]

Specifically, there are four key questions you need to ask and answer as you move toward choosing one or more software packages:

1. What kind of computer user am I?
2. Am I choosing for one project or for the next few years?

3. What kind of project(s) and database(s) will I be working on?
4. What kinds of analyses am I planning to do?

In addition to these four key questions, there are two cut-across issues to bear in mind:

- How important is it to you to maintain a sense of "closeness" to your data?
- What are your financial constraints when buying software and the hardware it needs to run on?

With these basic issues clear, you will be able to look at specific programs in a more active, deliberate way, seeing what does or does not meet your needs. (You may find it helpful to organize your answers to these questions on a worksheet, such as the one proposed in Weitzman & Miles, 1995b, which has rows for each of the questions below and columns for answers, implications/notes, and candidate programs.) Work your way from answering questions, to the implications of those answers for program choice, to candidate programs. For example, if you are working on a complex evaluation study, with a combination of structured interviews, focus groups, and case studies, you will need strong tools for tracking cases through different documents. You might find good support for this in a program's code structures, or through the use of speaker identifiers that track individuals throughout the database (see Question 3, below). Such suggestions are elaborated below.

Question 1: What Kind of Computer User Are You?

Your present level of computer use is an important factor in choice of a program. If you are new to computers, your best bet is probably to choose a word-processing program with advice from friends and begin using it, learning to use your computer's operating system (e.g., MS-DOS, Windows, or Mac) and getting comfortable with the idea of creating text, moving around in it, and revising it. That would bring you to what we'll call Level 1. Or you may have gotten acquainted with several different programs, use your operating system easily, and feel comfortable with the idea of exploring and learning new programs (Level 2). Or you may be a person with active interest in the ins and outs of how programs work (Level 3) and feel easy with customization, writing macros, and the like.

(I will not deal here with the "hacker," a Level 4 person who lives and breathes computing.)

Being more of a novice does not mean you have to choose a "baby" program, or even that you shouldn't choose a very complex program. It does mean, however, allowing for extra learning time, perhaps placing more emphasis on user-friendliness, and finding sources of support for your learning, such as friends or colleagues, or on-line discussion groups on the Internet. People at different levels seem to have quite different reactions to the same programs. So, for example, a person at Level 2 or 3 might like a program that puts the maximum information on one screen because this allows her to find what she wants quickly, and she might learn the program very quickly. A person at Level 1 might find all that information overwhelming at first, and might take a little longer to learn that program because of it. But, once he has learned the program, our Level 1 person would probably benefit from the layout in the same way as the more advanced computer user.

Question 2:
Are You Choosing for One Project or for the Next Few Years?

A word processor does not care what you are writing about, so most people pick one and stick with it until something better comes along and they feel motivated to learn it. But particular qualitative analysis programs tend to be good for certain types of analyses. Switching will cost you learning time and money. Think about whether you should choose the best program for this project or the program that best covers the kinds of projects you are considering over the next few years. For example, a particular code-and-retrieve program might look adequate for the current project and be cheaper or look easier to learn than some other program. But if you are likely to need a more fully featured code-based theory builder down the road, it might make more sense to get started with one of those now (assuming you choose one that includes good code-and-retrieve capabilities).

Question 3:
What Kind of Database and Project Will You Be Working On?

Here the questions begin to get a bit more specific. As you look at detailed software features, you need to play them against a series of de-

tailed issues. Because of the nature of computers, it becomes essential to give careful attention to the issue of understanding the nature and structure of qualitative data sets. The issues here have to do with the physical and logical form of the data: how structured and how consistent it is, how data about a case are organized, and so on. In terms of the issues presented below, there may be great variation from project to project, even within a given analytic approach (say, grounded theory, ethnography, or narrative analysis). Epistemological issues, such as the interpretive nature of observational notes, coding, or memos, or the social construction of interview data, although very important for research methodology, do not come into play here; the question is whether the program you choose provides the organizational tools for the text, graphics, audio, or video you want to put into it.

Data sources per case: single versus multiple. You may be collecting data on a case from many different sources (say your case is defined as a student, and you talk with several teachers, the student's parents and friends, the principal, and the student herself). Some programs are specifically designed to handle data organized like this, others are not designed this way but can handle multiple sources pretty well, and some really do not have the flexibility you'll need. As mentioned above, you should look for strong tools for tracking cases through different documents. Some programs provide good support for this in code structures (particularly programs with highly structured code systems, like NUD•IST and NVivo, and to lesser degrees in programs with flexible code systems like ATLAS/ti) or through the use of speaker identifiers in programs like AFTER, the Ethnograph, or Code-A-Text. Also, look for programs that are good at making links, such as those with hypertext capability, and that attach "source tags" telling you where information is coming from.

Single versus multiple cases. If you have multiple cases, you usually will want to sort them out according to different patterns or configurations, and/or work with only some of the cases, and/or do cross-case comparisons. Multicase studies can get complicated. For example, your cases might be students (and you might have data from multiple sources for each student). Your students might all be "nested" in (grouped by) classrooms, which might be nested within schools, which in turn might be nested in districts. Look for software that will easily select different portions of the database, and/or do configurational analysis (Ragin, 1987) across your

cases; software that can help you create multiple-case matrix displays, usually by gathering together the data that correspond to the different cells of the matrix, is also useful.

Fixed records versus revised. Will you be working with data that are fixed (such as official documents or survey responses) or data that will be revised (with corrections, added codes, annotations, memos, and so on)? Some programs make database revision easy, whereas others are quite rigid, so that revising can use up a lot of time and energy. Some will let you revise annotations and coding easily, but not the underlying text, and some will let you revise both. Although this has been a constraining issue up to now, the trend in new programs and upcoming revisions of existing ones is toward programs that allow you to edit underlying text easily.

Structured versus open. Are your data strictly organized (for example, responses to a standard questionnaire or interview) or free-form (running field notes, participant observation, and so on)? Highly organized data can usually be more easily, quickly, and powerfully managed in programs set up to accommodate them—for example, those with well-defined "records" for each case and "fields" (or variables) with data for each record. Structured surveys may benefit from survey-oriented programs like Sphinx Survey or EZ-Text, which take advantage of predictable structure to provide good data-manipulation tools. Free-form text demands a more flexible program. There are programs that specialize in one or the other type of data, and some that work fairly well with either.

Uniform versus diverse entries. Your data may all come from interviews, or you may have information of many sorts: archival documents, field observations, questionnaires, pictures, audiotapes, videotapes (this issue overlaps with single versus multiple sources, above). Some programs handle diverse data types easily, and others are narrow and stern in their requirements. If you will have diverse entries, look for software designed to handle multiple sources and types of data, with good source tags and good linking features in a hypertext mode. The ability to handle "off-line" data—referring you to material not actually loaded into your program—is a plus. Many programs can be tricked into doing this if you are clever about it. If you want to be able to code, and then retrieve, audio or video, look for programs like AFTER, ATLAS/ti, Code-A-Text, and InterClipper, which let you treat these media much like text.

Size of database. A program's database capacity may be expressed in terms of numbers of cases, numbers of data documents (files), size of individual files, and/or total database size, often expressed in kilobytes (K) or megabytes (MB). (Roughly, consider that a *single*-spaced page of printed text is about 2 to 3K.) Estimate your total size in whatever terms the program's limits are expressed, and at least double it. Most programs today are generous in terms of total database size. A few are still stingy when it comes to the size of individual texts. For example, in some programs when a document goes beyond about 10 pages, the program will insist on breaking it into smaller chunks or will open it only in a "read-only mode" browser.[4]

Question 4:
What Kind of Analyses Are You Planning to Do?

As mentioned above, identifying the name of your analysis methodology really won't do the trick here. Your choice of software depends on how you expect to go about analysis. This does not mean a detailed analysis plan, but a general sense of the style and approach you are expecting, which in turn will tell you the kinds of things you will need to be able to do with the data. For an excellent overview of a range of approaches to qualitative data analysis, and a discussion of some of the procedures associated with them, see Fielding and Lee (1998, chap. 2).

If you will be coding your data, you need a program that will let you code the way your methodology requires. If you are doing narrative analysis you may need to track temporal or narrative structures in certain ways. Or you may be focusing on building a web of hypertext links as a way of understanding the phenomena in your data, and your needs may be better served by one or another way of creating and representing that hypertext web. Coding is probably the best-supported approach at the current writing, and many researchers who use other approaches may find that their best option is to use a "coding" system for their own purposes—for example, to mark up the narrative structure of a text. More on this in the section headed "The Future," below.

The subsections below, laying out the parts of Question 4, should help you move beyond just the name of your methodology. They should help you identify the specific analytic moves you will need to make; the specific operations you will need to perform; the kinds of insights, inferences, and interpretations you will need to record; and the manner in which you plan to record them. In other words, they should help you to get specific about

the things you would do if working with paper and to translate these into the functions you will need from software.

Exploratory versus confirmatory. Are you mainly planning to poke around in your data to see what they are like, evolving your ideas inductively? Or do you have some specific hypotheses in mind linked to an existing theory to test deductively? If the former, it is especially important that you have features of fast and powerful search and retrieval, easy coding and revision, along with good text and/or graphic display.

If, on the other hand, you have a beginning theory and want to test some specific hypotheses, programs with strong theory-building and -testing features are better bets. Look for programs that test propositions, or those that help you develop and extend conceptual networks.

Coding scheme firm at start versus evolving. Does your study have a fairly well defined a priori scheme for codes (categories, key words), perhaps theoretically derived, that you will apply to your data? Or will such a scheme evolve as you go, in a grounded theory style, using the "constant comparative" method (Glaser, 1965; Glaser & Strauss, 1967; Strauss & Corbin, 1998)? If the latter, it is especially important that you have easy on-screen coding (rather than being required to code on hard copy, or having to deal with cumbersome on-screen coding procedures) and features supporting easy or automated revision of codes. Hypertext linking capabilities are helpful here too. "Automated" coding (in which the program applies a code according to a rule you set up, such as when a certain phrase or a combination of other codes exists) can be helpful in either case.

Multiple versus single coding. Some programs let you assign several different codes to the same segment of text, including higher-order codes, and may let you overlap or nest coded "chunks" (the ranges of text you apply codes to). Others are stern: one chunk, one code. Still other programs will let you apply more than one code to a chunk, but will not "know" that there are multiple codes on the chunk—they'll treat it like two chunks, one for each code.

Iterative versus one pass. Do you want—and do you have the time—to keep walking through your data several times, taking different and revised cuts? Or will you limit yourself to one pass? An iterative intent should point you toward programs that give you a good display of your previous

coding, are flexible, invite repeated runs, make coding revision easy, have good search and autocoding features, allow you to track connections between different parts of the text with hypertext, and can make a log of your work as you go. (See also the question of whether your records are fixed or revisable during analysis.)

Fineness of analysis. Will your analysis focus on specific words? Or lines of text? Or sentences? Paragraphs? Pages? Whole files? Look to see what the program permits (or requires, or forbids) you to do. How flexible is it? Can you look at *varying* sizes of chunks in your data? Can you define free-form segments with ease? Some programs make you choose the size of your codable segments when you first import the data, whereas others let you mix and match chunk sizes as you go.

Interest in context of data. When the program pulls out chunks of text in response to your search requests, how much surrounding information do you want to have? Do you need only the word, phrase, or line itself? Do you want the preceding and following lines/sentences/paragraphs? Do you want to see the entire file? Do you need to be able to jump right to that place in the file and do some work on it (e.g., code, edit, annotate)? Do you want the information to be marked with a "source tag" that tells you where it came from (e.g., Interview 3 with Janice Chang, page 22, line 6)? Or do you just want the source information without the text itself? Programs vary widely on this.

Intentions for displays. Analysis goes much better when you can see organized, compressed information in one place rather than in page after page of unreduced text. Some programs produce output in list form (lists of text segments, hits, codes, and so on). Some can help you produce matrix displays. They may list text segments or codes for each cell of a matrix, although you will have to actually arrange them in a matrix for display. Look for programs that let you edit, reduce, or summarize hits before you put them into a text-filled matrix with your word processor. Some programs can give you quantitative data (generally frequencies) in a matrix. Others can give you networks or hierarchical diagrams, the other major form of data display.

Qualitative only or numbers included. If your data, and/or your analyses, include the possibility of number crunching, look to see whether the

program will count things and/or whether it can share information with quantitative analysis programs such as SPSS or SAS. Think carefully about what kind of quantitative analysis you'll be doing, and make sure the program you are thinking about can arrange the data appropriately. Consider, too, whether the program can link qualitative and quantitative data in a meaningful way (in terms of the analytic approach you are taking). For example, do you need to be able to select subsets of your qualitative data based on quantitative scores or demographics? Or do you need to use your qualitative coding to generate scaled variables for statistical analysis in SPSS? Or do you want to be able to generate word or code frequency tables for statistical analysis?

Collaboration. If you will be working with a team and more than one of you will be working on data analysis, look to see how the program supports collaboration. Some are fine if you just want to divide up the work with each of you coding different parts of the data and then combining the work. Others will support comparing multiple researchers' interpretations of the same data. Some programs will allow multiple users to access a shared database over a network; others will allow you to merge periodically separate copies of the database that different researchers have been working on. Programs differ in how much control they give you over the merge process. Some allow you to specify what the program will do if it finds, say, codes or memos with the same name in each of the copies being merged, whereas others follow a fixed rule. Some programs are good at letting you tell which copy a code, memo, or other object came from; others lose all identifying information so you have to use tricks like using different names in each copy (e.g., I might start the names of all codes I create with my initials, and you start yours with your initials). Some programs offer specific features for letting you compare the coding of two different researchers, for example, by showing you a table in which you can see the coding done by each.

Cut-Across Issues

The two main cut-across issues are closeness to the data and financial resources. Let's dispense quickly with the latter question first. Software varies dramatically in price. The range of prices for the programs we reviewed in Weitzman and Miles (1995b) was $0 to $1,644 per user. That is

still the range. (Look for discounts for educational users and multiple-user "site licenses.") In addition, programs vary a lot in the hardware they require to run efficiently. You obviously cannot use a program if it is too expensive for you, if it requires a machine you cannot afford, or if it runs on the wrong platform—say, PC instead of Mac. Happily, the U.S. Centers for Disease Control now distributes two programs free via the Web: EZ-Text for qualitative surveys and AnSWR for unstructured text. Also happily, reports from the field are that the Macintosh computers being sold today (the G3 is today's top Mac) run even the most powerful new PC programs satisfactorily with PC "emulation" software. This will, presumably, continue to be the case with future generations of Macs.

The remaining issue, closeness to the data, is more complex, and I also address it below in the section headed "Debates in the Field." For choice purposes, remember to think about what *kind* of closeness to the data is important to you. Many researchers fear that working with qualitative data on a computer will have the effect of "distancing" them from their data. This can in fact be the case. You may wind up looking at only small chunks of text at a time, or maybe even just line-number references to where the text is. This is a far cry from the feeling of deep immersion in the data that comes from reading and flipping through piles of paper.

But other programs minimize this effect. They typically keep your data files onscreen in front of you at all times; show you search results by scrolling to the hit, so that you see it in its full context; and allow you to execute most or all actions from the same data-viewing screen. Programs that allow you to build in hypertext links between different points in your data, provide good facilities for keeping track of where you are in the database, display your coding and memoing, and allow you to pull together related data quickly can in some ways help you get even *closer* to the data than you can with paper transcripts. If you choose with this consideration in mind, software can *help,* rather than hinder, your work at staying close to the data. It can, in fact, help keep you from drowning in those piles of paper.

However, having software that enhances the sense of closeness to the data may not be a crucial issue for everyone. Some researchers do not mind relying heavily on printed transcripts to get a feeling of closeness, whereas others think such heavy reliance defeats the purpose of QDA software. Furthermore, some projects simply do not require intense closeness to the data. You may be doing more abstract work, and in fact may *want* to move away from the raw data.

◆ Debates in the Field

A number of debates have taken place over the past two decades in the qualitative research community about whether the use of software is a good idea and, if so, what kinds of software are a good idea. I will address four of the debated issues here: closeness to the data, whether software drives methodology, whether new researchers should start off doing analysis by hand, and whether software really affects rigor, consistency, and thoroughness.[5]

Closeness to the Data

The issue of closeness to the data, which I have just discussed in terms of program choice, has been one of the big concerns raised by qualitative researchers over the years. Experienced researchers have often found that as difficult as it was, the process of spending endless hours sitting on the floor surrounded by piles and piles of paper led them, by necessity, to a very rich and thorough familiarity with their data. But as I have tried to argue above, software need not cut down on this familiarity. Software neither makes it better nor worse, it simply changes it. Although some programs still create the sense that you are staring at just a small window of text with no sense of what lies around it, there are now many programs available that provide rich contextual information (such as source information, graphical maps of hypertext links, navigable outlines, and linked lists of codes, documents, and text segments), and may in fact help you get to know your data better than ever before.

Does Software Drive Methodology?

Another concern has been that researchers might wind up adapting their research to the software they use, rather than the other way around (Coffey et al., 1996; Kelle, 1997; Lonkila, 1995)—that is, that the software will impose a methodological or conceptual approach. In fact, software developers bring assumptions, conceptual frameworks, and sometimes even methodological and theoretical ideologies to the development of their products. These have important implications for the impact that using a particular program will have on your analyses. However, as I have argued elsewhere, you need not, and in fact should not, be trapped by these assumptions and frameworks; there are often ways of bending a

program to your own purposes (see Weitzman, in press). For example, a program may allow you to define only hierarchical relations among codes. You might work around this by creating redundant codes in different parts of the hierarchy, or by keeping track of the extra relationships *you* want to define with memos and network diagrams.

The fact that developers bring conceptual assumptions to their work is in fact one of the strengths of the field. Many of the developers, particularly of code-and-retrieve and code-based theory builder programs, are researchers themselves. They have invested enormous intellectual energy in finding the right tools for analyses of different types, and the user can benefit greatly from their investment.

It is also true that the design of the software can have an impact on analysis. For example, different programs work with different "metaphors"—that is, different ways of presenting the relationships among codes, and between codes and text. Kwalitan, NUD•IST, the Ethnograph, and winMAX all allow hierarchical relations among codes. For studies in which you are organizing your conceptual categories hierarchically, these programs offer significant strengths. If you want to represent nonhierarchical relationships, even if you choose to try to work around this metaphor, it may be less comfortable than using a program like ATLAS/ti that explicitly supports more flexible networks. HyperRESEARCH emphasizes the relationship between codes and cases, rather than codes and chunks of text. When you code a chunk of text, you create an entry on what looks like an index card for a particular case. This strongly supports and encourages thinking that stresses casewise and cross-case phenomena, but makes it harder to look for and think about relationships among codes *within* a text (although version 2, under development at the time of this writing, appears to be solving this problem). Finally, Code-A-Text offers a quite different set of coding metaphors: (a) codes arranged into "scales" (you can assign only one code from each scale to a segment of text, useful if you want to code your text chunks by making mutually exclusive judgments on a variety of factors), (b) codes automatically assigned according to words in the text, and (c) open-ended "interpretations" you write about each text chunk. Working with this collection of coding metaphors could be expected to lead you to consider your text in somewhat different ways than with one of the other metaphors described above.

Similar issues exist in the choice of other types of software, such as textbase managers. InfoTree32 XT allows you to arrange texts in a hierarchical tree and lets you drag texts around to rearrange them. Folio Views

also allows you to create a hierarchical outline, but gives you the additional capability of creating multiple, nonsequential "groupings" of texts that you can activate any time you wish. Folio Views and askSam let you insert fields whenever and wherever you want in any given record, whereas InfoTree32 XT requires that you use a standard field set (of your own design) throughout the whole database. Idealist not only lets you customize the field set for each record, you can also create a variety of record types, each with its own set of fields, and mix them in the same database. Finally, whereas most of these programs show you just one record at a time, Folio Views shows you a word processor-like view in which records appear one right after the other as paragraphs. Clearly, each of these programs shows you a quite different view of your data, and so each may encourage different ways of thinking about your data. The different programs all have different strengths and weaknesses. It is also true that a clever user will be able to bend each of these flexible packages to a wide variety of different tasks, overcoming many of the differences between them.

Each of these assumptions is both a benefit for some modes of analysis and a constraint. The key, then, is not to get trapped by the assumptions of the program. If you are aware of what they are, you can be clever and work around them. The program should serve *your* analytic needs, goals, and assumptions, not the other way around. Researchers interested in empirical work on the impacts of different programs on research are again encouraged to refer to Fielding and Lee (1998).

Should New Researchers Start Off Doing Analysis by Hand?

There is no clear-cut answer to this question. Certainly, it is important that new researchers begin by learning about how to do good analysis, rather than just how to use a program. Whether that means doing a first project by hand or learning about analysis and software to do it with in the same course is a question best left to teachers. I have taught both ways, and in my experience students benefit from having some experience with manual methods, if only a few coding exercises, so that they can get the feel of what is happening analytically before they start worrying about using the software.

Does Software Really Affect Rigor? Consistency? Thoroughness?

Some researchers are dedicated to the notion that software makes for more rigorous research. There are even rumors floating around of federal funding agencies requiring the use of software in grant proposals. Yet, as I have argued above, software will not pull good work out of a poor researcher. On the other hand, for all the reasons outlined above in the subsection headed "Real Hopes," software can in fact help competent researchers do more rigorous, consistent, and thorough analysis than they otherwise might. The issue should be conceptualized not as whether the software makes the work more rigorous, but whether the researcher uses the software to do more rigorous work than he or she could without it.

◆ The Future

It is my hope that the future will see a continuation of current trends, both in scholarship and in software development. Some of my specific hopes are outlined below.

Needs for Scholarship on the Topic

Ongoing review work. In addition to books like Weitzman and Miles (1995b) and its upcoming revision (which I am coauthoring with Nigel Fielding and Ray Lee), which offer comprehensive comparative reviews of the range of software available at a particular time, there is a need for regularly appearing reviews of new and revised programs as they appear. The journal *Field Methods* (formerly *Cultural Anthropology Methods*) offers regular software reviews (of quantitative as well as qualitative programs) in the same way that many journals feature regular book reviews. More journals that serve qualitative research audiences should follow this lead.

Debate on methodological questions. The kind of controversial issues addressed in this chapter need to be subjected to continued debate in the literature and among researchers. We need to be both wary of unintended influences of software and actively participating in shaping the future development of software by arguing (constructively) with developers about what we need and what we do not like.

More empirical work. The kind of empirical work on the impact of software on analysis that has been pioneered by Fielding and Lee (1998), Weaver and Atkinson (1995), Horney and Healey (1991), and Walker (1993) needs to be continued. Opinions about the impact of software are nice, but we also need to continue to subject our hypotheses to empirical research.

Needs for Software Development

We can at this point identify some of the needs of researchers that are not yet met. For example, the field is still lagging in its support for case-oriented work. A few programs have features built in for explicitly tracking individual cases through multiple documents, but few programs are set up with a strong case-oriented structure.

Display building, especially of matrices, still needs much development. A product newly released at the time of this writing, NVivo (from the developers of NUD•IST), allows you to build an interactive matrix in which you can click on cells to call up the corresponding text. Matt Miles's dream of a program that would combine this sort of functionality with the ability to actually compose the summary text for the output matrix (rather than switching to a word processor) is still one step away.

Tools for narrative and discourse analysis are still lagging as well. Researchers using these approaches continue to call for features that let them flexibly describe the structure of text and discourse, and longitudinal researchers do not yet have much in the way of tools built explicitly for tracking cases over time, though NVivo has an "attributes" feature that allows you to attach date values to codes or documents.[6] In each of these cases, researchers can either adapt coding systems to their needs or look for yet other kinds of software (such as hypertext authoring programs, project schedulers, and so on) that they can adapt to their needs.

Finally, because no one program will ever do it all best, researchers need developers to create the possibility of importing and exporting marked-up, coded, annotated data from one program to another. At this writing, there is just a little of this beginning to happen. The developers of Code-A-Text, the Ethnograph, and winMAX have agreed to work on a common structure, partially realized at this point. And ATLAS/ti has become the first program to support export of fully developed projects in XML, a new markup language that may succeed HTML, the World Wide Web formatting standard. A common standard like this, if adopted by other developers, would allow researchers to move fully developed projects easily from one program to another, just as we can now move tabular data among multiple spreadsheet and database programs.

◆ Conclusion

Unlike the situation just a decade or so ago, qualitative researchers now have available to them an array of very good software tools to assist in their research, and the use of software—including, but not limited to, word processors—seems more and more to be a regular part of the qualitative research process. There is still no one "best" program, not even for a particular methodology, and that's good. It means that researchers have to think through their methods and choose programs that fit, which should keep them from becoming reliant on the software to lead them. As researchers continue to hunt around for programs that will do the things they want, and do them better, software developers will likely continue to respond by making their programs more and more useful.

What else can we hope will come out of this collaboration between users and developers in the near future? More and better tools for sharing analyses and raw data, perhaps by allowing posting of project databases, with analytic markups, links, and memos, to the World Wide Web, as ATLAS/ti allows, or on CDs; tools for building complex reports that include analyses and data right in the report itself; and more and better tools for supporting collaboration among research teams, and for involving informants in the research process without intensive computer training.

◆ Notes

1. I discuss some exceptions in the subsection below headed "False Hopes and Fears."

2. The first release of NVivo lets you draw diagrams, but any connections you draw are represented only in the diagram, they are not representations of the defined relationships among codes and other objects, as in ATLAS/ti. You see the actual relationships among codes in a hierarchical "explorer" with expandable and collapsible branches, as in NUD•IST, the Ethnograph, and winMAX.

3. This section does not contain much in the way of references to specific software, both because the landscape changes every few years and because a single chapter does not allow for responsible comparisons among programs.

4. For any warning like this, check at the time you are choosing to see if the program under consideration presents this problem. This type of problem is worked at so regularly by developers that it would be unfair and unhelpful for me to name particular programs. Things change.

5. The reader interested in pursuing these questions further is referred to Fielding and Lee (1998) for reports of users' experiences of these and other issues when using different programs.

6. You can, in fact, attach not only date values, but text or numerical values as well.

◆ References

Bogdan, R. C., & Biklen, S. K. (1982). *Qualitative research for education: An introduction to theory and methods.* Boston: Allyn & Bacon.

Coffey, A., & Atkinson, P. (1996). *Making sense of qualitative data: Complementary research strategies.* Thousand Oaks, CA: Sage.

Coffey, A., Holbrook, B., & Atkinson, P. (1996). Qualitative data analysis: Technologies and representations. *Sociological Research Online, 1*(1). Available Internet: http://www. socresonline.org.uk/socresonline/1/1/4.html

Drass, K. A. (1980). The analysis of qualitative data: A computer program. *Urban Life, 9,* 322-353.

Fielding, N. G., & Lee, R. M. (Eds.). (1991). *Using computers in qualitative research.* London: Sage.

Fielding, N. G., & Lee, R. M. (1998). *Computer analysis and qualitative research.* London: Sage.

Glaser, B. G. (1965). The constant comparative method of qualitative analysis. *Social Problems, 12,* 436-445.

Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research.* Chicago: Aldine.

Goetz, J. P., & LeCompte, M. D. (1984). *Ethnography and qualitative design in educational research.* New York: Academic Press.

Horney, M. A., & Healey, D. (1991, April). *Hypertext and database tools for qualitative research.* Paper presented at the annual meeting of the American Educational Research Association, Chicago.

Kelle, U. (Ed.). (1995). *Computer-aided qualitative data analysis: Theory, methods and practice.* London: Sage.

Kelle, U. (1997). Theory building in qualitative research and computer programs for the management of textual data. *Sociological Research Online,* 2(2). Available Internet: http://www.socresonline.org.uk/socresonline/2/2/1.html

Lee, R. M., & Fielding, N. G. (1991). Computing for qualitative research: Options, problems and potential. In N. G. Fielding & R. M. Lee (Eds.), *Using computers in qualitative research* (pp. 1-13). London: Sage.

Lofland, J., & Lofland, L. H. (1984). *Analyzing social settings: A guide to qualitative observation and analysis* (2nd ed.). Belmont, CA: Wadsworth.

Lonkila, M. (1995). Grounded theory as an emerging paradigm for computer-assisted qualitative data analysis. In U. Kelle (Ed.), *Computer-aided qualitative data analysis: Theory, methods and practice* (pp. 41-51). London: Sage.

Mangabeira, W. (Ed.). (1996). Qualitative sociology and computer programs: Advent and diffusion of CAQDAS [Special issue]. *Current Sociology, 44*(1).

Miles, M. B., & Huberman, A. M. (1984). *Qualitative data analysis: A sourcebook of new methods.* Beverly Hills, CA: Sage.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook* (2nd ed.). Thousand Oaks, CA: Sage.

Ragin, C. C. (1987). *The comparative method: Moving beyond qualitative and quantitative strategies.* Berkeley: University of California Press.

Seidel, J. V., & Clark, J. A. (1984). The Ethnograph: A computer program for the analysis of qualitative data. *Qualitative Sociology, 7,* 110-125.

Shelly, A., & Sibert, E. (1985). *The QUALOG users' manual.* Syracuse, NY: Syracuse University, School of Computer and Information Science.

Strauss, A. L., & Corbin, J. (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (2nd ed.). Thousand Oaks, CA: Sage.

Tesch, R. (1990). *Qualitative research: Analysis types and software tools.* New York: Falmer.

Tesch, R. (1991). Computers and qualitative data II. *Qualitative Sociology, 14*(3).

Walker, B. L. (1993). Computer analysis of qualitative data: A comparison of three packages. *Qualitative Health Research, 3*(1), 91-111.

Weaver, A., & Atkinson, P. (1995). From coding to hypertext: Strategies for microcomputing and qualitative data analysis. In R. G. Burgess (Ed.), *Studies in qualitative methodology.* Greenwich, CT: JAI.

Weitzman, E. A. (1999). Analyzing qualitative data with computer software. *Health Services Research, 34*(5), 1241-1263.

Weitzman, E. A., & Miles, M. B. (1995a). Choosing software for qualitative data analysis: An overview. *Cultural Anthropology Methods, 7*(1), 1-5.

Weitzman, E. A., & Miles, M. B. (1995b). *Computer programs for qualitative data analysis: A software sourcebook.* Thousand Oaks, CA: Sage.